GRAPHQL

& beyond

@felipesoares_

why

how

when

@felipesoares6_

felipesoares6

sorocabacss

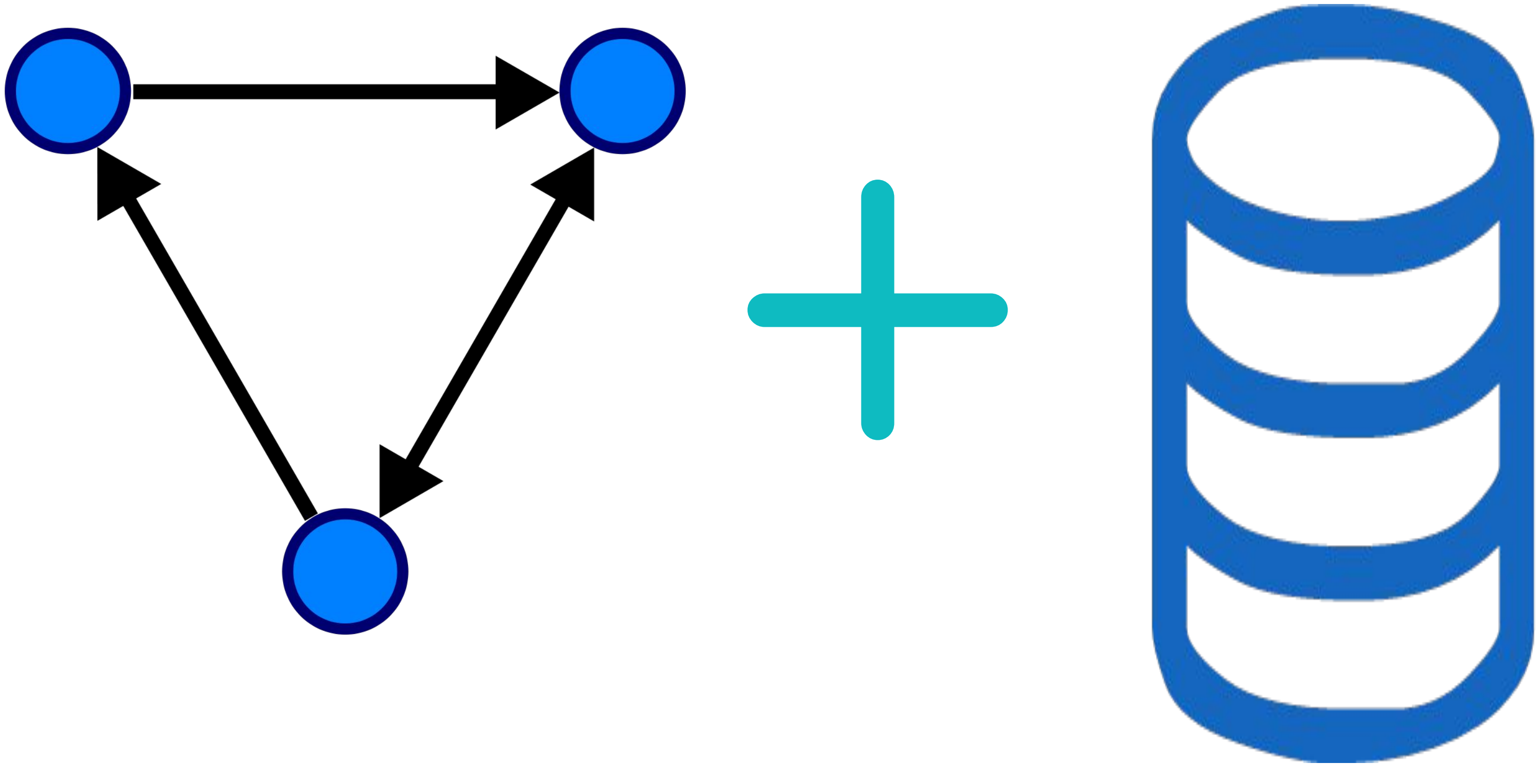
CODE MINER

NOSSO BAR
RED BAR E LANCHONETE
BRAHMA
CORRAJ
Copy House

PADARIA
REAL
Desde 1957

GraphQL is a query language

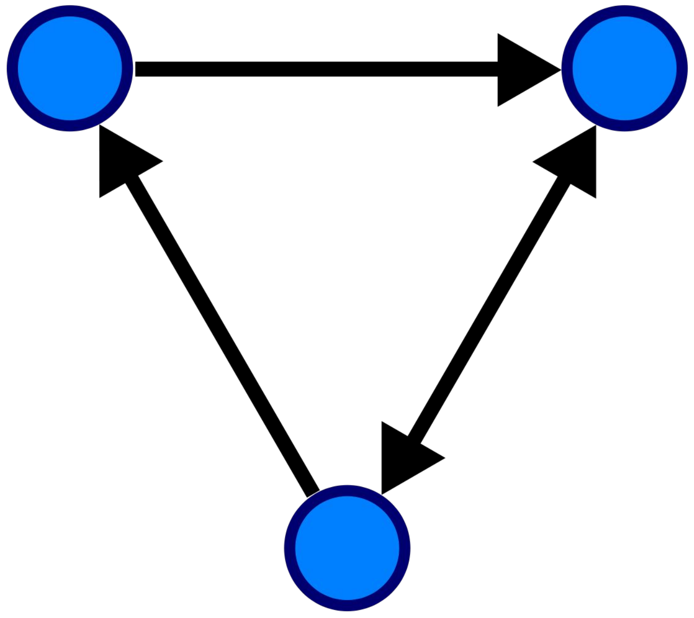
& alternative to REST



graphs

-> nodes

-> edges



query language

```
SELECT *

FROM
```

```
type House {
  edge: Person
}
```

```
type Person {
  ...
}
```

why?

3 reasons

get just what you want

enables declarative
data fetching

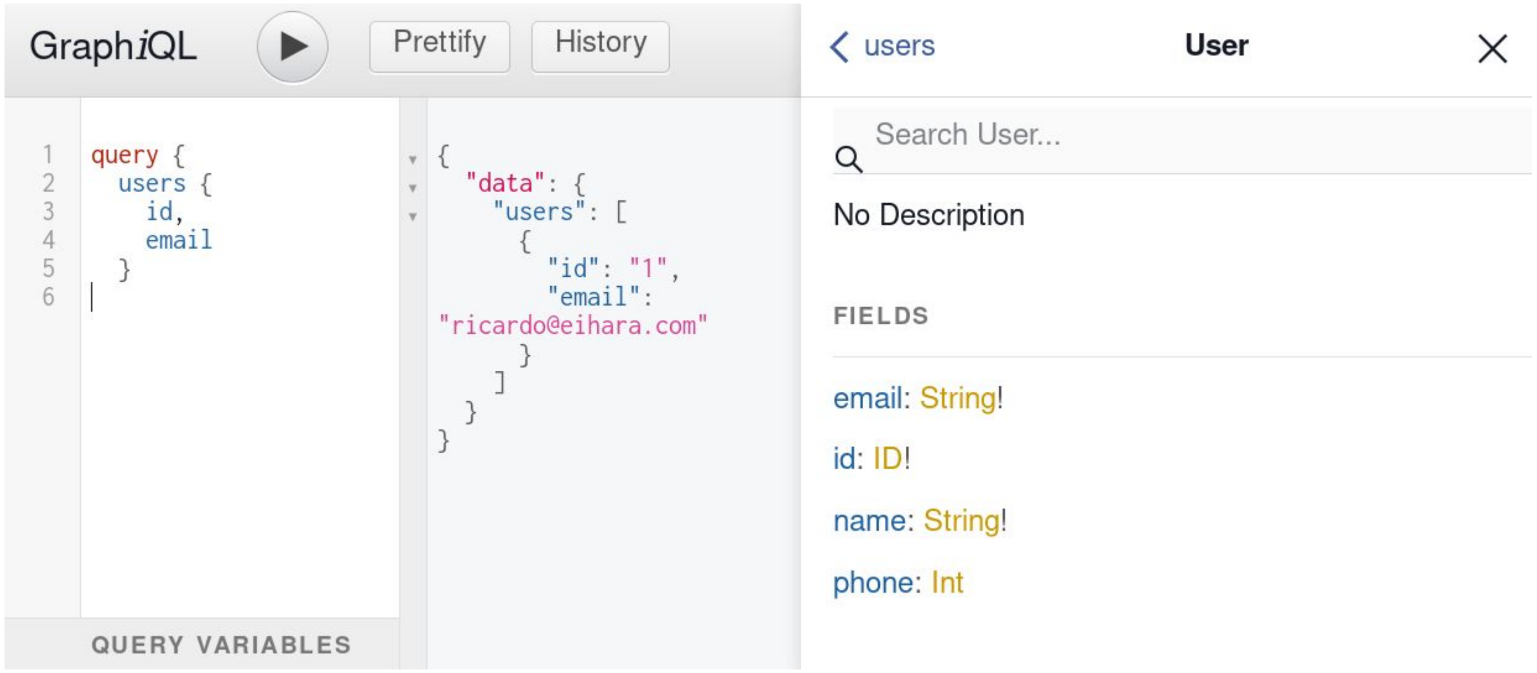
~~overfetching~~

~~underfetching~~

nice and smooth
contracts

+ typed schemas
+ docs

playground



GraphiQL | Prettify | History

```
query {
  users {
    id,
    email
  }

```

```
{
  "data": {
    "users": [
      {
        "id": "1",
        "email":
"ricardo@eihara.com"
      }
    ]
  }
}
```

< users — User

Search User...

No Description

FIELDS

email: String!

id: ID!

name: String!

phone: Int

QUERY VARIABLES

clients everywhere

only exposes a single endpoint

consume the same endpoint in != ways

how?

```
-> schema
   -> queries
   -> mutations
      -> resolvers
         -> responses
```

schema

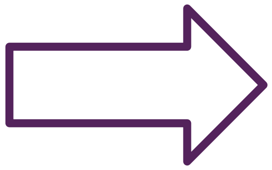
```
type Query {
    users: [User!]!,
    user: User!
}
```

```
type User {
    id: ID!,
    name: String,
    email: String
}
```

```
schema
     -> queries
     -> mutations
        resolvers
           responses
```

queries

GET /users



```
query {
  users {
    id,
    name,
    email
  }
}
```

mutations

POST /users

```
mutation {
    createUser(
        name: “Felipeson”,
        email: “felipe@son.com”
    ) {
        id
    }
}
```

```
schema
    queries
    mutations
          -> resolvers
               responses
```

resolvers

```
field :users do
  resolve -> (users) { User.all }
end

field :user do
  resolve -> (_, args) { User.find(args[:id]) }
end

???: { 'api.com/top' }
```

schema

queries

mutations

resolvers

-> responses

responses

loading

```
query {
    users {
        id,
        name,
        email
    }
}
```

$\Rightarrow$

```
"data": {
    users: [
        {
            id: "1",
            name: "Felipe Soares",
            email: "felipe@son.com"
        }
    ]
}
```

```
query {
    users {
        id,
        name,
        gender
    }
}
```

```
"errors": [
    {
        "message": "error",
        "location": ... ,
        "fields": ...
    }
]
```

how how

server-side

bit.ly/2FQPwRG

```
gem 'graphql'
```

```
Root = GraphQL::Schema.define do
  mutation(Types::MutationType)
  query(Types::QueryType)
end
```

```
Types::QueryType =
GraphQL::ObjectType.define
  do
    name 'Query'

     ...
end
```

```
field :users, !types[Types::UserType] do
  resolve ->(_obj, _args, _ctx) { User.all }
end
```

client-side

bit.ly/2zRBNUq

apollo

query

```
import gql from ‘graphql-tag’

import { Query } from ‘react-apollo’
```

```
const GET_USERS = gql`
  query {
    users {
      id,
      name,
      email
    }
  }`
```

```
const GET_USERS = gql`
  query {
    users {
      id,
      name,
      email
    }
  }`
```

```
<Query query={GET_USERS}>
    {({ load, error, data }) => {

        ...

    }
</Query>
```

when?

is difficult to close
the API contracts

~~overfetching~~

~~underfetching~~

consume the same endpoint in != ways

consuming != APIs

learn something new!

thanks!

felipesoares6

@felipesoares6_

bit.ly/2FQPwRG
client-side

bit.ly/2zRBNUq
server-side